SPECIFICATION AMENDMENT

Please amend the paragraph located at page 2, line 22 through page 3, line 2 as follows:

The present invention provides a system and method, or framework, for declaratively specifying authorization enforcement points and associating them with permission subclasses. Authorization enforcement code is inserted into one or more of the target classes of the target application according to the authorization enforcement points.

Please amend the paragraph located at page 4, lines 5 through 7 as follows:

Fig. 1 is a block diagram illustrating the overall structure of some embodiments of the present invention implemented using the ~~Java~~ JAVA™ Authentication and Authorization Service API (JARS) is shown;

Please amend the paragraph located at page 4, line 18 through page 5, line 2 as follows:

Preferred embodiments of the invention are now described with reference to the drawings. In accordance with the invention, and with reference to Fig. 1, a block diagram illustrates the overall structure of some embodiments implemented using the ~~Java~~ JAVA™ Authentication and Authorization Service API (JARS) The JARS provides flexible and scalable mechanisms for securing client-side and server-side ~~Java~~ JAVA™ applications. The JARS is pluggable and stackable, and lets developers incorporate standard security mechanisms like Solaris NIS, Windows NT, lightweight access directory protocol (LDAP), Kerberos, and others, into applications in a consistent, configurable way.

Please amend the paragraph located at page 5, lines 3 through 9 as follows:

The system of the present invention directly instruments target class binaries at deployment time or at run time through a class loader based on deployment descriptor settings. Instrumentation of ~~Java~~ JAVA™ bytecode comprises inserting a short sequence of bytecode at designated points in the ~~Java~~ JAVA™ classes of an application. Such a technique is commonly used to facilitate runtime analysis of those classes, which involves mostly profiling, monitoring, or other introspection tasks. However, the present invention uses the technique to insert its instrumentation code (100 in Fig. 1).

Please amend the paragraph located at page 5, lines 10 through 16 as follows:

There are two types of instrumentation, static and dynamic. Static instrumentation of code can occur during or after compilation, but dynamic instrumentation can take place at runtime, which includes after the target application is loaded in a memory for execution. Runtime class instrumentation is performed through a class preprocessor 50, whereby a class preprocessor 50 inserts instrumentation code 100 at the required places in the target ~~Java~~ JAVA™ classes 102 just before it is loaded by a ~~Java~~ JAVA™ virtual machine. The class preprocessor 50 works in conjunction with an instrumentation class loader 106.

Please amend the paragraph located at page 8, lines 1 through 13 as follows:

With reference to Fig. 5, an example of the instrumented code 102b class is shown according to the example implementation of the invention of Figs. 2 and 3, which is the state of the method code 102a from Fig. 2 after instrumentation of instrumentation code 100 by the class preprocessor 50. As mentioned in the comment section on lines 4-5

of the code, although the authorization enforcement code in the example is actually inserted as byte code, for the purposes of illustration, the Java code version of the instrumented code 102b is shown in FIG. 5. As can be seen, in the instrumented part of code 102b, the preprocessor 50 has read the deployment descriptor code 104a, and added the appropriate instrumentation code 100 according to the parameters in the deployment descriptor code 104a. The instrumentation code added to the target class code 102a in this example 102b creates permissionFactory and privilgedActionFactory objects, sets the properties needed to read the profile database by running the ~~a~~ setPropery method of those objects, and executes a getPrivilegeAction method which executes the ReadAction privilege action of the instrumentation code 100.

Please amend the paragraph located at page 15, lines 1 through 6 as follows:

ABSTRACT OF THE DISCLOSURE

~~Disclosed are systems and methods for providing an authorization framework for applications.~~ A framework is provided for declaratively specifying authorization enforcement points and associating them with classes and subclasses by using declarations that map constants, local variables, or instance variables to permission classes and subclasses such as Java Permission, PermissionFactory, PrivilegedAction and PrivilegedActionFactory classes.